****

基于α-β剪纸算法

的五子棋实现

|  |  |
| --- | --- |
| 姓名: | 赵元川 |
| 学号: | 10225101427 |

目录

[1、引言 1](#_Toc183954566)

[1.1 背景 1](#_Toc183954567)

[1.2 五子棋中的布局和策略 1](#_Toc183954568)

[1.3人工智能在五子棋中的应用 2](#_Toc183954569)

[1.4任务与分工 2](#_Toc183954570)

[2、理论背景 3](#_Toc183954571)

[2.1 α-β剪枝算法 3](#_Toc183954572)

[2.1.1 α-β剪枝算法简介 3](#_Toc183954573)

[2.1.2 α-β剪枝的工作原理 3](#_Toc183954574)

[2.1.3 α-β剪枝的优势 4](#_Toc183954575)

[2.2评估函数与启发式搜索 4](#_Toc183954576)

[2.2.1 评估函数的作用 4](#_Toc183954577)

[2.2.1 启发式搜索 4](#_Toc183954578)

[3、系统架构与实现 5](#_Toc183954579)

[3.1 技术栈 5](#_Toc183954580)

[3.2 系统架构与运行逻辑 5](#_Toc183954581)

[3.3 α-β剪枝算法实现 7](#_Toc183954582)

[3.3.1 基本流程 7](#_Toc183954583)

[3.3.2 具体实现步骤 8](#_Toc183954584)

[3.4 评估函数实现与启发式搜索 9](#_Toc183954585)

[4、项目实现效果 10](#_Toc183954586)

1、引言

1.1 背景

五子棋（Gomoku），又叫五子连珠，是一种源自中国的传统棋类游戏，通常在15×15或19×19的棋盘上进行。两位玩家轮流在棋盘上放置黑白棋子，目标是在一条直线上（横、竖或斜）连接五颗同色棋子。五子棋规则简洁、易学，但其战略性与复杂度却随着棋局的进行迅速增加，因此成为了棋类爱好者和人工智能研究者的研究热点。

随着计算机科学的发展，五子棋逐渐成为人工智能（AI）研究的一个重要领域。研究者采用各种算法来模拟人类玩家的思维过程，优化决策过程。其中，α-β剪枝算法作为一种高效的博弈树搜索算法，广泛应用于各类棋类游戏的AI开发。通过减少不必要的搜索，α-β剪枝能够在搜索树中快速剔除不可能的选择，极大提升了决策效率。虽然五子棋的搜索空间相对较小，但通过合适的优化和策略，AI依然能在复杂的局面下做出最优决策。

除了算法的选择，五子棋中的各种布局和策略也是AI决策的关键因素。在实际对局中，玩家不仅需要简单地追求五连珠，还必须考虑到棋盘上的各种形势变化、对手的策略以及潜在的威胁。为了应对这些复杂情况，AI需要掌握多种布局和战术，包括冲三、死三、活三等常见的局面。

1.2 五子棋中的布局和策略

五子棋的布局可以分为进攻性和防守性两大类，其中进攻性布局是指通过自己的棋子形成有利的棋型，逼迫对方防守，甚至直接威胁到胜利。防守性布局则是指在对方形成进攻威胁时，通过巧妙的布局和堵塞手段进行反制。以下是一些常见的五子棋布局，它们在实际对弈中具有重要意义：

1. 冲三：冲三是指在棋盘上形成一个连续的三颗棋子的局面，其中两侧为空或只有对方棋子阻挡。冲三通常用作进攻性布局，通过在一侧不断扩展棋子来威胁对方。然而，冲三的弱点在于如果被对方堵住其中一个空位，便无法继续扩展，进攻势头将被遏制。
2. 死三：死三是指三颗连续的棋子被对方棋子限制，导致无法在一侧继续扩展以形成五连珠。死三是五子棋中最常见的陷阱之一，如果不及时防范，可能会导致局面的被动。玩家往往通过将死三转化为活三或四连来改变局势。
3. 活三：活三是指在三颗棋子的两侧没有被对方棋子封堵，玩家可以在任意一侧继续扩展棋子形成五连珠。活三是五子棋中最强的布局之一，因为它不仅能威胁对方，还具备很高的扩展性。在实际对局中，活三通常意味着即将构成五连珠，因此需要格外小心。
4. 四连：四连指四颗相连的同色棋子，其中一侧存在空位。四连是五子棋中的一种强势布局，因为它距离五连珠仅一步之遥。在实际游戏中，如果一方形成了四连，另一方必须立即进行防守，否则对方很容易借机取得胜利。

这些布局和策略不仅决定了玩家在局中的进攻和防守能力，还影响着整个棋局的进展。对于AI来说，识别并充分利用这些布局，是保证其战术效果和胜率的关键。

1.3人工智能在五子棋中的应用

五子棋AI的设计主要面临两大挑战：首先是如何通过搜索算法快速找到最优的落子位置，其次是如何在复杂的局面中准确识别并利用棋盘中的各种布局。为了应对这些挑战，研究人员和开发者使用了多种策略和算法。

最基本的方式是使用极大极小算法，该算法模拟了两位玩家在每个可能的局面下选择最优行动的过程。然而，**极大极小算法**存在一个显著问题：其搜索树非常庞大，随着棋盘尺寸的增加，计算量呈指数级增长。为了解决这个问题，**α-β剪纸算法**应运而生。α-β剪枝通过在搜索过程中剔除掉一些不必要的分支，大大减少了需要评估的节点数，从而提升了计算效率。

尽管如此，五子棋AI的决策不仅仅依赖于简单的局部搜索，还需要具备一定的战略性。为了提升AI的决策质量，许多AI系统都引入了**评估函数**，用来衡量棋盘局面的好坏。评估函数通常结合了棋盘上棋子的分布情况、各类布局的形成可能性、对方的威胁程度等因素，通过量化的方式帮助AI选择最佳的落子位置。

1.4任务与分工

本项目设计并实现一个基于α-β剪枝算法的五子棋AI系统。具体而言，本项目的任务包括：

1. 算法实现：使用α-β剪枝算法进行搜索和决策，优化搜索效率，避免无谓的计算。
2. 布局识别：设计并实现能够识别冲三、死三、活三等布局的评估函数，使AI能够根据不同的棋局采取合适的进攻或防守策略。
3. 用户界面：开发一个用户友好的图形界面，允许玩家与AI进行对弈，同时能够实时显示棋盘状态。
4. 搜索优化：对AI的博弈树搜索进行进一步的优化，提高AI的棋力和响应能力。

小组分工如下：

赵元川：负责算法设计实现和布局识别，包括评估函数和运行时的游戏主逻辑。具体为main.cpp，GobangAI类，GameBoard类。此实验报告中的图均为赵元川画制。

张力骏：负责用户界面和搜索优化。具体为AhoCorasick类，MainWindow类，Point类以及Board类。

2、理论背景

2.1 α-β剪枝算法

2.1.1 α-β剪枝算法简介

α-β剪枝算法是一种优化的极大极小（Minimax）算法，常用于博弈树的搜索中，尤其适用于两人零和博弈问题，如五子棋、国际象棋等。在标准的极大极小算法中，搜索树的每个节点表示一个可能的游戏状态，算法通过模拟两方玩家的决策过程，寻找最优的落子位置。然而，由于极大极小算法需要遍历树中所有可能的节点，其计算量会随着树的深度和分支数量急剧增加，导致效率低下。

α-β剪枝通过在搜索过程中有效地“剪掉”一些不必要的分支，显著减少了搜索树的节点数，从而提高了搜索效率。具体来说，α-β剪枝算法在极大极小算法的基础上引入了两个参数：α和β，分别表示当前节点的最大下界和最小上界。通过比较当前节点的评估值与子节点的评估值，α-β剪枝能够在某些情况下避免对无关分支的评估。

2.1.2 α-β剪枝的工作原理

在极大极小算法中，玩家会选择使自己获利最大或使对方损失最小的动。α-β剪枝算法在此基础上引入了如下两个剪枝规则：

1. α（最大化玩家的最优解）：表示当前最大化玩家已知的最好的选择。即，在极大玩家的回合中，α值会记录当前已知的最大值，代表在当前节点及其父节点下，最大化玩家的最优解。
2. β（最小化玩家的最优解）：表示当前最小化玩家已知的最好的选择。即在极小玩家的回合中，β值会记录当前已知的最小值，代表在当前节点及其父节点下，最小化玩家的最优解。

通过这些值的不断更新，α-β剪枝能够在搜索过程中判断哪些分支没有必要再继续扩展，从而减少了不必要的计算。

2.1.3 α-β剪枝的优势

α-β剪枝的主要优势在于显著减少了搜索的树节点数，从而提高了搜索效率。在最理想的情况下，α-β剪枝能够使极大极小算法的时间复杂度从 **O(bd)** 降到 **O(bd/2)**，其中 **b** 是每个节点的最大子节点数，**d** 是搜索树的深度。

然而，实际效果取决于剪枝顺序。如果能够先访问最优的分支，那么α-β剪枝的效果会更好，搜索效率会更高。

2.2评估函数与启发式搜索

2.2.1 评估函数的作用

为了让AI能够有效判断当前局面的优劣，需要设计一个评估函数。评估函数通过分析棋盘上的各个位置、棋子的分布和布局情况，给出一个数值表示该局面对当前玩家的有利程度。评估函数通常会考虑以下几个因素：

1. 连续棋子的数量
2. 对方威胁
3. 棋盘的占据情况

评估函数的设计通常是五子棋AI中的关键部分，合适的评估函数能够使AI根据当前棋局做出合理的决策。本项目中由于算力有限仅考虑了前两个因素。

2.2.1 启发式搜索

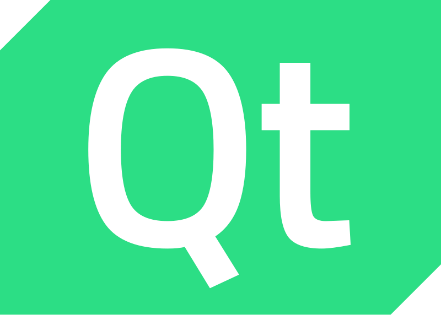
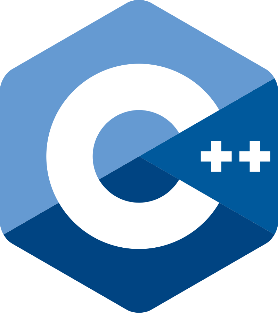
启发式搜索是一种结合了评估函数的搜索策略，常用于博弈类问题。在五子棋AI中，启发式搜索通常通过α-β剪枝算法对博弈树进行搜索，通过剪枝减少计算量，同时利用评估函数对局面进行估值，指导搜索过程。

启发式搜索结合了博弈树搜索和局面评估，使得AI能够在较短时间内找到较优的决策，避免了暴力搜索所有可能的棋局。

3、系统架构与实现

3.1 技术栈

本项目使用了Qt作为了UI界面编写的框架，使用了C++作为了编程语言实现了具体算法。

3.2 系统架构与运行逻辑

本项目使用了面向对象思想将任务合理分配给了各种类。首先，**GobangAI** 类负责实现五子棋的人工智能部分。它通过评估函数来分析当前棋盘的局势，并利用α-β剪枝算法优化搜索过程，从而高效地决定AI的最佳走法。这使得AI在面对复杂局面时能够快速做出决策，同时最大化其获胜概率。

与此相对，**Board** 类则管理着整个棋盘的状态。它通过一个二维数组记录棋盘上每个位置的状态，包括空位、黑棋和白棋。它不仅提供落子功能，还能够实时检测是否有玩家已经形成了五连珠从而获得胜利。此外， **Board** 类还负责打印棋盘，帮助开发者调试和查看当前局势。

**Point** 类则表示单个的“动作”或“落子”，它包含了落子的坐标以及棋子的颜色。每次AI或玩家做出决策时，都会生成一个Point实例，表示在棋盘上进行的具体操作。这个类还包含评估值，用来衡量该落子的优劣。

**AhoCorasick**类为AC自动机，负责棋子分数的匹配。用于算法搜索优化部分。

最后，**GameBoard** 类是整个游戏的控制中心。它负责管理整个游戏的流程，包括轮流机制、胜负判断以及与用户界面的交互。**GameBoard** 类确保每一步落子都符合五子棋的规则，并能够实时判断当前局势，判断是否有人获胜，或者游戏是否结束。这个还包含了玩家的棋子的颜色，并负责决策过程。对于AI玩家，决策是由 **GobangAI** 类来执行的，而对于人工玩家，则是用户输入来进行控制。

这几个类通过各自的功能相互协作，构成了一个完整的五子棋游戏系统。GobangAI 提供智能决策，Board 管理棋盘状态，Point 记录每次操作，AhoCorasick优化搜索，而 GameBoard 控制整个游戏的流程和规则。

部分类图1－1所示。



图1－1 类图

Figure 1-1 Class Diagram

系统运行流程如图2－1所示。



图2－1 流程图

Figure 2-1 Flow Diagram

3.3 α-β剪枝算法实现

3.3.1 基本流程

α-β剪枝算法的目标是通过模拟两位玩家（AI和对手）交替落子的过程来决定当前局势下最优的走法。具体来说，AI通过极大化自己的得分（Maximizing Player），而对手则通过最小化AI的得分（Minimizing Player）来做出决策。以下是算法的核心步骤：

1. 节点评估：在每个节点，AI通过评估函数评估当前局势的优劣。评估函数返回一个数值，表示当前局面对AI的有利程度。这个数值越大，意味着局势越有利于AI。
2. 极大极小搜索：算法通过递归的方式在游戏树中搜索每一层的所有可能的棋局。对于AI的回合，算法尝试最大化当前局面的评估值，而对于对手的回合，算法尝试最小化AI的评估值。
3. 剪纸操作：在搜索过程中，α和β值的更新将帮助算法决定哪些分支可以被剪去。具体来说，如果当前节点的评估值已经无法影响最终决策（即α值大于或等于β值），那么就可以剪掉这个分支，从而避免不必要的计算。

3.3.2 具体实现步骤

3.3.2.1 初始化与参数定义

在α-β剪枝中，我们需要定义一个递归函数alphaBetaPlay，其核心参数包括：state(当前棋局) isBlack(当前玩家的棋子颜色), depth(当前搜索深度), alpha(当前搜索路径的最大值，初始值为负无穷), beta(当前搜索路径的最小值，初始值为正无穷), maxdepth(最大搜索深度)。

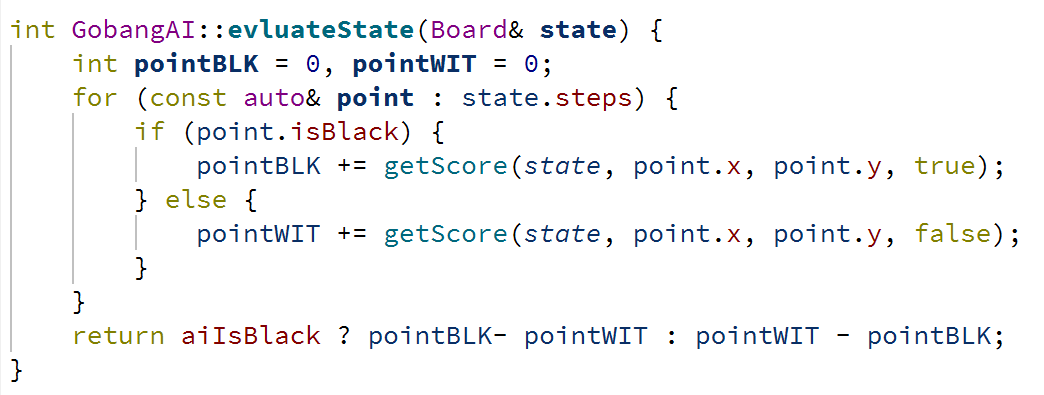
3.3.2.2 递归搜索过程

递归函数alphaBetaPlay的核心思想是通过探索博弈树的各个分支，寻找最优的走法。每个节点代表一种棋盘状态，算法尝试选择一个最优的分支进行搜索，直到达到预定的搜索深度或找到胜利条件。但搜索所有的状态空间非常耗时耗算力且不理想，所以这里使用了启发式搜索，搜索了所有棋子的邻居作为下一步的落子。

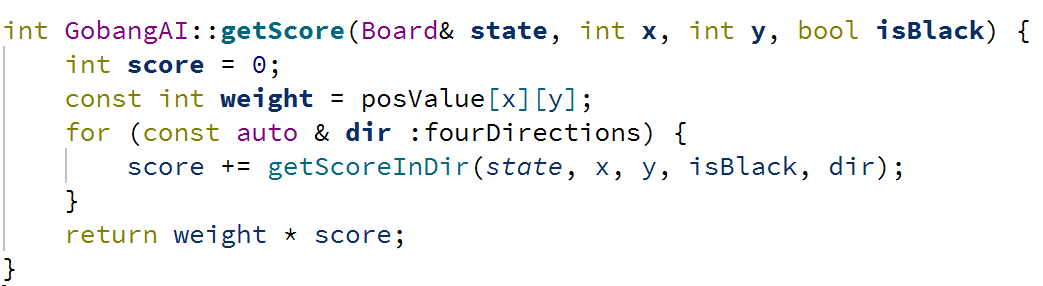


3.4 评估函数实现与启发式搜索

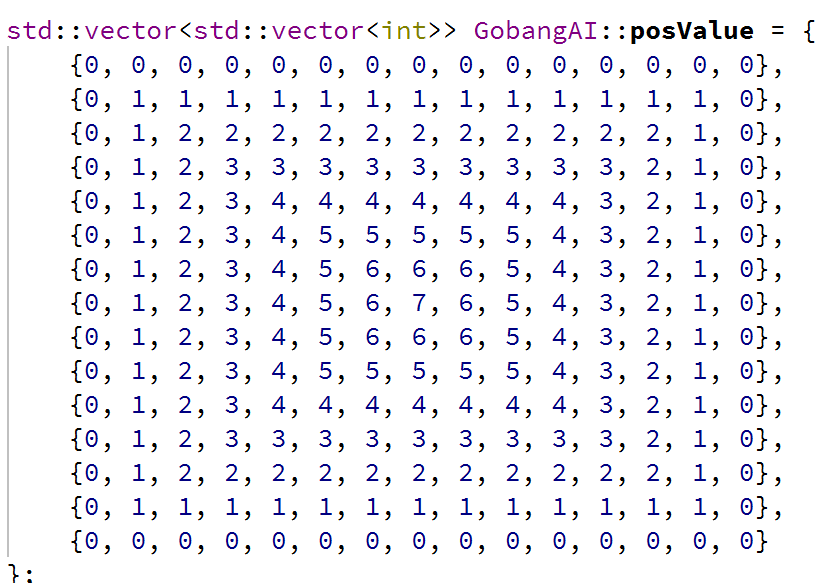
评估函数是α-β剪枝算法中的关键，它根据当前棋盘的状态计算出一个数值，表示该局面对AI的有利程度。评估函数会根据五子棋的规则判断是否形成了五连珠、四连、活三等重要的布局，并根据这些布局赋予不同的分数。例如，活三和四连等对AI有利，死三和对方的连珠则会被赋予负分。



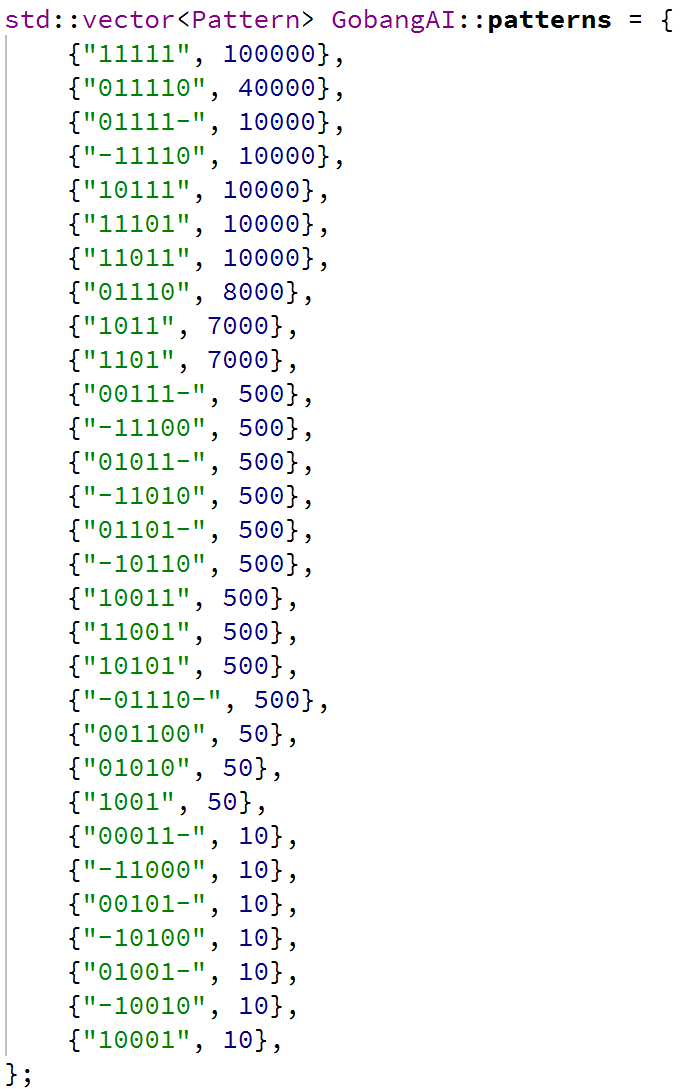
这里的getScore函数会获取这个点的分数，这个函数会遍历8个方向来获取棋子字符产然后使用AC自动机匹配棋子布局对应的分数，最后进行相加。实现如下。



这里使用到的权重如下（接近中间部分的棋子有更高的胜率）：



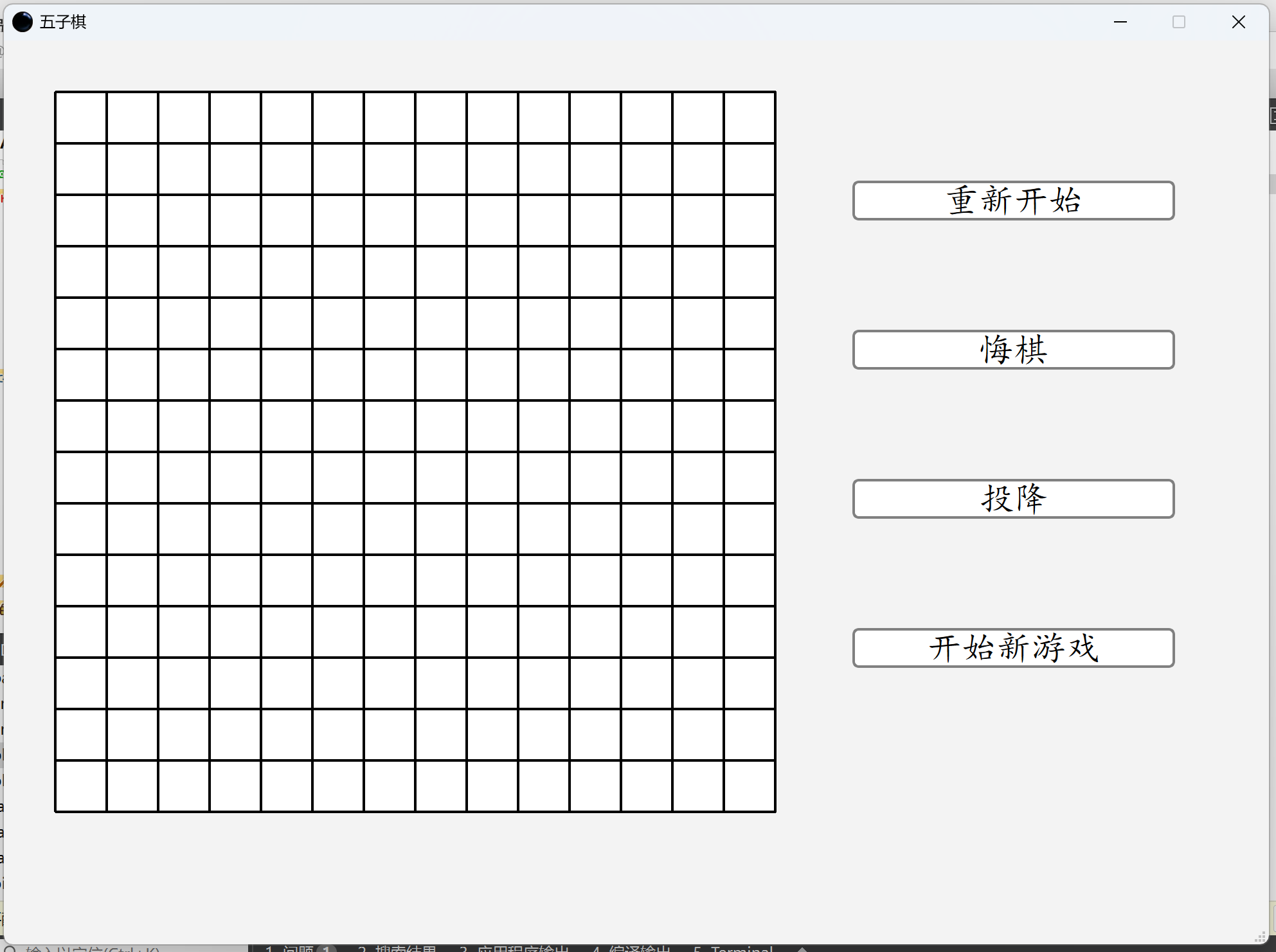
搜索使用到的模式匹配如下，1代表我方棋子，0代表对方棋子，-代表无棋子。

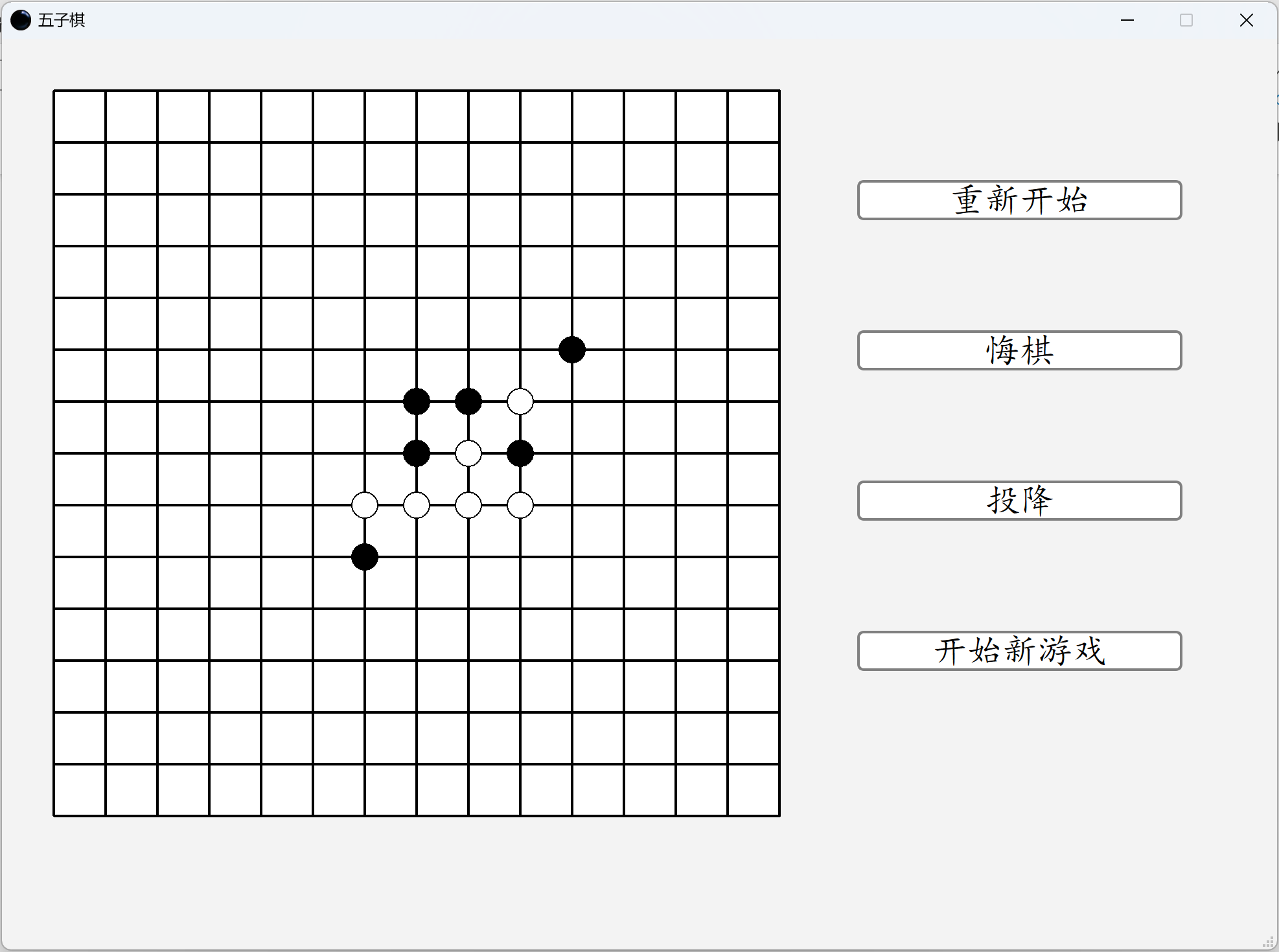


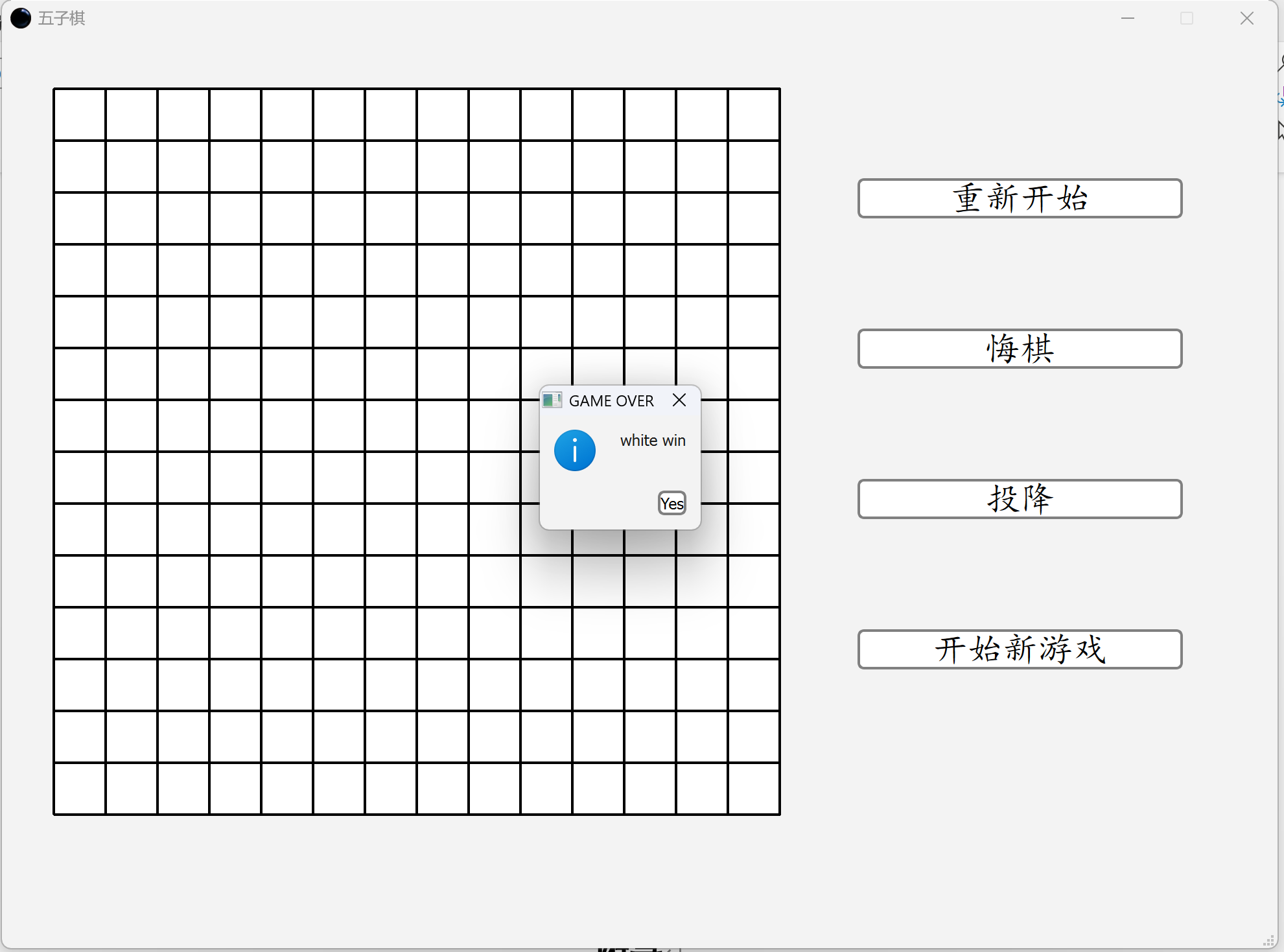
4、项目实现效果

本项目实现了AI五子棋，实现了高棋力的同时(能做到三子相连的时候进行及时的杀棋和拦截)，实现了高响应速度。游戏效果如下图：









本项目在Github上进行了开源，欢迎进行下载游玩：

https://github.com/WhizzK/AIGobang